EXHIBIT E

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

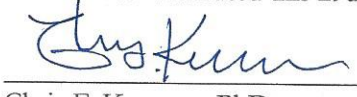
ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	)	Civil Action No. 10-03561 WHA

REPLY EXPERT REPORT OF CHRIS F. KEMERER, Ph.D.

February 29, 2016

CONFIDENTIAL – ATTORNEYS' EYES ONLY
PURSUANT TO PROTECTIVE ORDER

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed this 29th day of February, 2016, in Pittsburgh, PA.



Chris F. Kemerer, PhD
Chris F. Kemerer PhD LLC

TABLE OF CONTENTS

I. Assignment

1. I have been asked to respond to certain opinions and assertions in the expert reports of Dr. Owen Astrachan and Dr. Gregory Leonard by evaluating the nature and purpose of both the Java APIs and the Android system that draws on them, and to evaluate the benefits to Google from its commercial use of the Java APIs in Android.

2. In addressing these questions, I draw on my academic and professional background, which includes more than 30 years of experience in the areas of software engineering, technology adoption and diffusion, the creation and management of information systems, and the market economics of software and information systems.

3. I am being compensated for my work on this case at a rate of $595 per hour. My compensation is not contingent upon my testimony or on the result of this proceeding. Appendix A to this report lists the materials I considered in preparing this report.

4. My work is ongoing, and I reserve the right to modify or supplement my conclusions as additional information becomes available to me, or as I perform further analysis.

II. Qualifications

5. My credentials are detailed in §II of my Initial Report, submitted on January 8, 2016. My full qualifications and complete CV are also presented in Appendix B of my Initial Report.

III. Summary of Opinions

6. Contrary to Dr. Astrachan's assertions, the stability analysis that I conducted regarding method change is well supported by pre-existing research and was carried out reliably and consistent with the prior research. The results of the stability analysis show that copying the 37 Java API packages contributed to the stability of the Google created packages in the Android core libraries. Also, Dr. Astrachan claims that deleted methods should be accounted for in the stability analysis. While I disagree with Dr. Astrachan in that regard, in order to respond to his assertion I also carry out the stability analysis in a manner that accounts both for changed methods and deleted methods. The results obtained from this additional analysis do not vary in any material way from the results of my prior analysis.

7. Contrary to Dr. Leonard's assertions that the "top" Android apps are purportedly written only in C/C++, my prior analysis of the top Android apps demonstrates that each of the top 100 Android apps to date uses, and depends significantly on, the 37 copied Java API packages. Further, contrary to Dr. Leonard's

assertions that the top Android apps are only written in C/C++, my analysis shows that the top apps put forward by Dr. Leonard contain a significant number of lines of Java code. Dr. Leonard provides no code analysis for his conclusions, and therefore they are unsupported.

IV. Stability Analysis

8. Dr. Astrachan comments that he does not understand the objective definition of stability of the APIs in Java and Android.[1] However, as explained in my initial report submitted on January 8, 2016 and in my fair use report submitted on February 8, 2016 (incorporated by reference here), stability was defined through the conservative measurement of changed methods alone.[2] An established body of literature supports the use of method declaration changes as an effective measure of stability in Java and Android APIs.[3] In particular, as discussed in my earlier report, the measure used is the equivalent of the "CEM" metric developed in a 2012 paper by Raemaekers *et al.* that uses method changes to measure software stability.[4] This measure of stability would not only capture changes that lead to loss of backward compatibility (a measure advocated by Dr. Astrachan[5]), but would further capture changes that are likely to lead the audience of developers to believe that APIs lack such compatibility or have other issues, as established in the literature addressed in my prior reports.

9. Two recent related research studies address these issues. Linares-Vasquez, *et al.* (2013), in a paper entitled "API Change and Fault Proneness: A Threat to the Success of Android Apps" study the fault and change-proneness of a sample of 7,097 apps. The authors conclude that making use of fault and change-prone APIs was negatively related to the success of those apps. In particular, they note that "the use of unstable APIs that undergo numerous changes in their interfaces can cause backward compatibility problems or require frequent updates to the app. Such updates, in turn, can introduce defects into the applications using the unstable APIs."[6] In a related study, Yuan Tian *et al.* (2015), in a paper entitled "What are the Characteristics of High-Rated Apps? A Case Study on Free Android Applications" investigate 28 possible factors to understand 1,492 high and low-rated apps. They find a high correlation[7] between the average number of methods changed in a particular Android API and the average number of bugs in that Android API.

[1] Astrachan Rebuttal Report, ¶ 171.
[2] *See e.g.* Initial Report, ¶¶ 100-107, Fair Use Report, ¶¶ 84-96.
[3] *See e.g.* Fair Use Report, ¶¶ 80-85.
[4] *See* Fair Use Report, ¶¶ 82, 85.
[5] Astrachan Rebuttal Report, ¶ 173.
[6] Linares-Vasquez et al., p. 2.
[7] They report it as "greater than 0.7", where 1.0 would be the theoretical maximum perfect correlation.

10. Notwithstanding my application of this standard analysis used to measure API stability, which considers only changed methods, Dr. Astrachan criticizes this approach suggesting that deleted methods should also be considered in the stability analysis.[8]

11. Despite the fact that the research literature supports changed methods as an effective measure of stability, in order to respond to Dr. Astrachan's assertions I consider the effect of deleted methods only as a sensitivity analysis. The results of this analysis are that, when considering deleted methods in addition to changed methods, the stability analysis results do not change in any material respect compared to my prior analysis, supporting my earlier decision not to include them.

12. As before, in order to understand the stability of the copied Java APIs compared to other Android APIs over all version releases in both Android and Java, I analyzed the publicly available method declarations in all publicly released versions of both Android and Java SE. I used the same data sets and methodologies for data extraction and parsing as discussed in my prior reports, which produces a set of tabular data documenting the evolution of method declarations in each API over time. However, in addition to measuring changed methods,[9] as set forth in my prior reports, I also considered deleted methods for both the Java APIs and the Android APIs. The number of deleted methods is counted based on the list of methods existing in continuous API levels. Any one particular method structure existing in one API level, but deleted in a subsequent API level, is counted as one method deletion in this analysis. A custom R script was used to calculate the number of changed methods and deletions for each package between all continuous API levels for both Android APIs and Java APIs.[10]

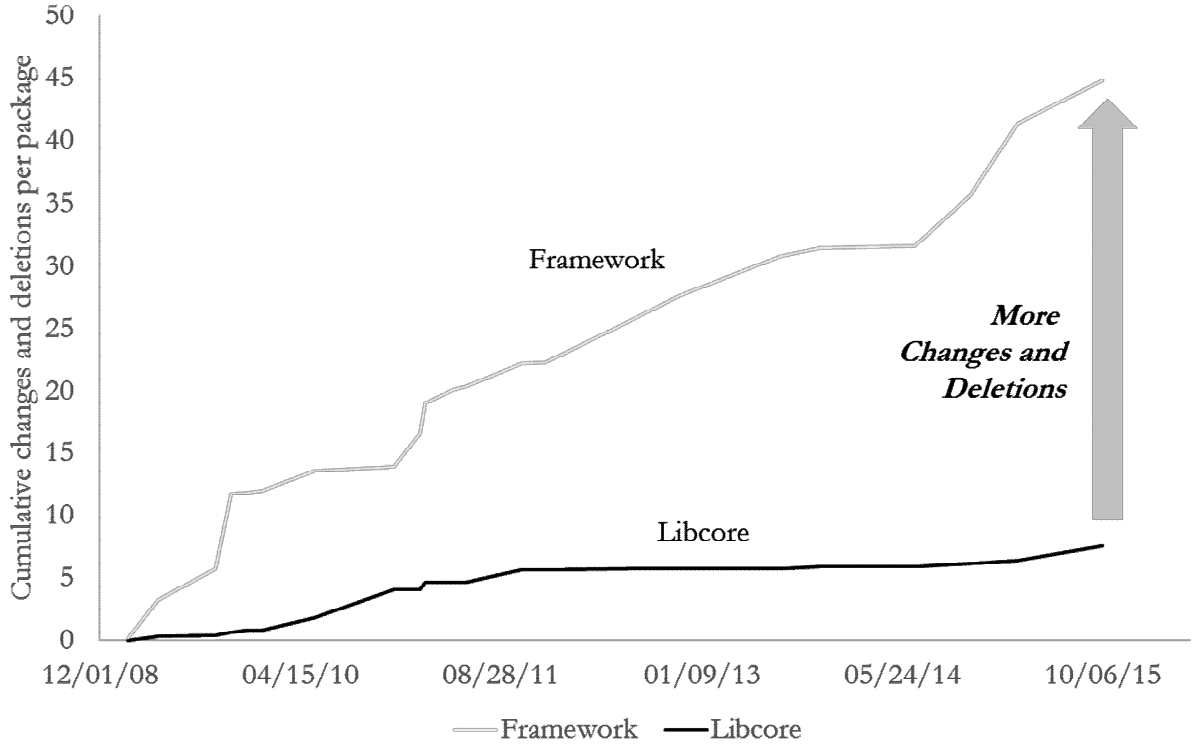
13. As before, stability (accounting both for changed methods and deletions) was considered, and evaluated for the Google categories of the Android APIs set forth in my prior reports (Libcore, Framework, External).[11] The Java APIs were subdivided into two different categories, based on whether they are included in the 37 infringed Java API packages or not.

14. Between two subsequent versions and across all versions, of both Android and Java APIs, the number of changed methods and deletions in each package was calculated. The result was then divided by the number of packages to obtain an average number of changes per package. The cumulative number of changes and deletions per package in each category was then graphed to visually track the stability of each

[8] Astrachan Rebuttal Report, ¶ 172.
[9] Method structure considered in assessing method change includes method abstract, method synchronization, method exception, method return type, method transient, method volatile, method value, method static, method final, method deprecated, method visibility, and method type. If any of the method structure changes in a method, then the method change is counted as 1.
[10] *See* Appendices C and D, respectively, for the R method-counting scripts used to calculate changes across different versions of the Java SE and Android platforms.
[11] Note that the relatively small number of "External" category APIs are not included in the analysis.

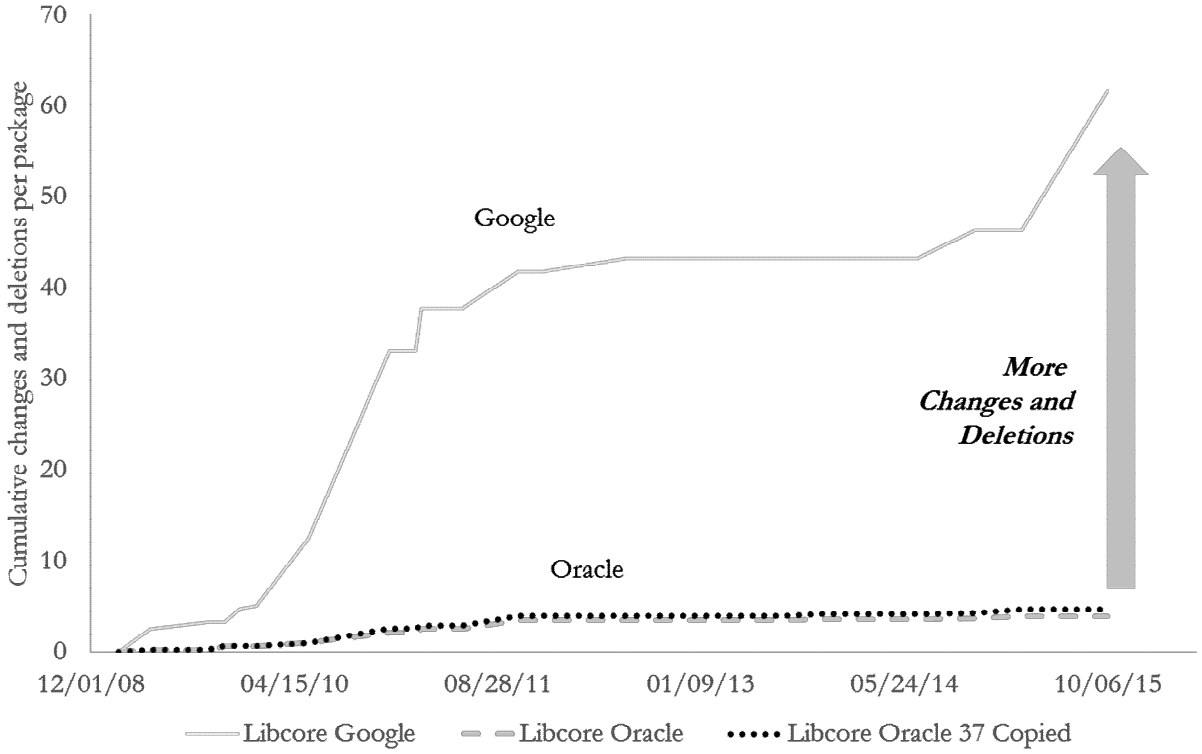
version over time. Within Android, the Libcore APIs, which are largely composed of Oracle's Java APIs, stabilized much earlier than the largely Google-based Framework APIs, and had far fewer cumulative changes and deletions per package across all API levels. The Libcore APIs stabilized after Android API Level 9, which was released in December, 2010. Conversely, the Framework APIs exhibits many more changes and deletions, and have yet to stabilize fully. (See Figure 1)

Figure 1: Cumulative Changes and Deletions per Package (Framework vs Libcore)



15. The more rapid stabilization of the Android Libcore APIs compared with its Framework APIs can be attributed to the presence of Oracle's Java APIs. Within the Libcore APIs, the Oracle APIs independently achieved stability very rapidly compared to the Google APIs. Moreover, their role in the stabilization of Android Libcore is evident when isolating their specific contributions with respect to stability. As illustrated in Figure 2, cumulative changes and deletions for both the Oracle Java APIs and specifically those from the 37 copied Java API packages in the Libcore category taper off beginning with API level 9, which was released in December, 2010. Meanwhile, Libcore Google APIs have many more cumulative changes and deletions per package across all API levels and apparently have yet to stabilize, even in the latest version. This suggests that the stability of Android Libcore is driven by the 37 copied Java API packages.

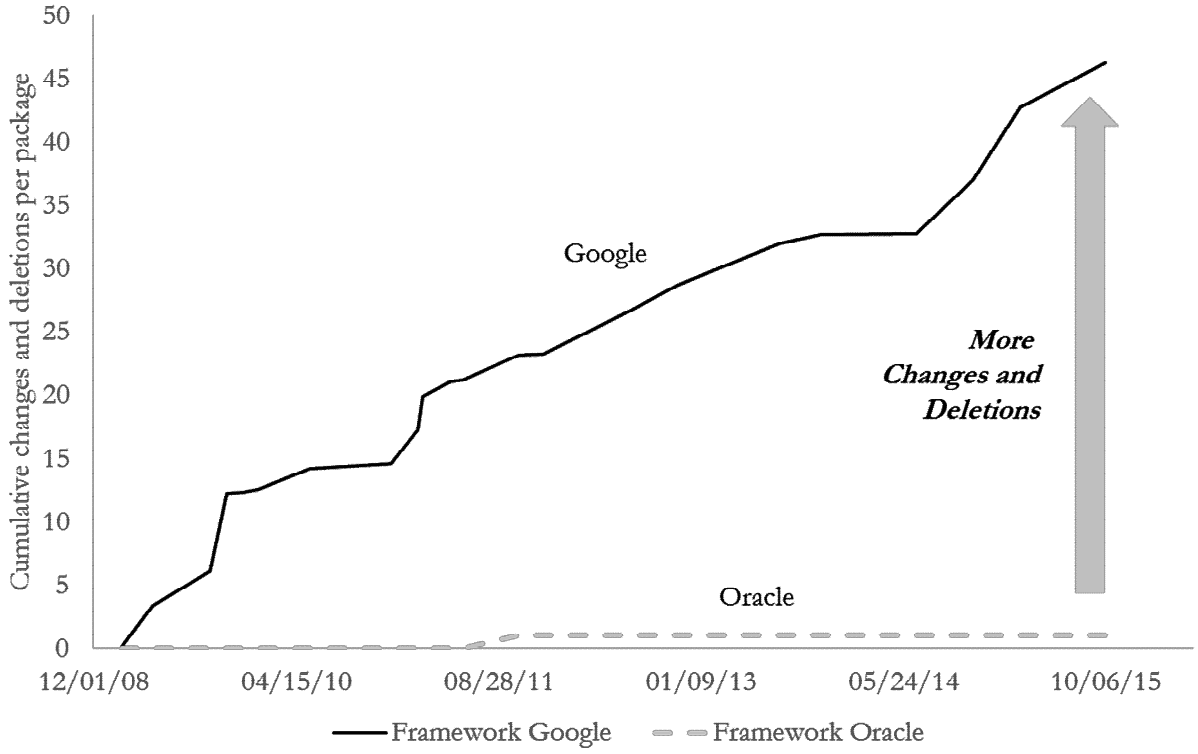
Figure 2: Cumulative Libcore Changes and Deletions per Package (Google vs Oracle)



16. The Android Framework, 98% of which is composed of Google APIs, took far longer to stabilize. There are only two Oracle APIs[12] in the Android Framework category, and there are only two changed methods or deletions for those two Framework Oracle APIs over all API levels. As a result, nearly all of the change volatility in the Android Framework APIs arises from Google APIs. As illustrated in Figure 3, the Framework Google APIs still have not demonstrated stabilization, even in the most recent API Level.

[12] These are javax.microedition.khronos.egl and javax.microedition.khronos.opengles. These are not two of the 37 copied Java APIs at issue, but are two other APIs not shown to be owned by Oracle, but which had their origin in the Java platform, instead of being created by Google.

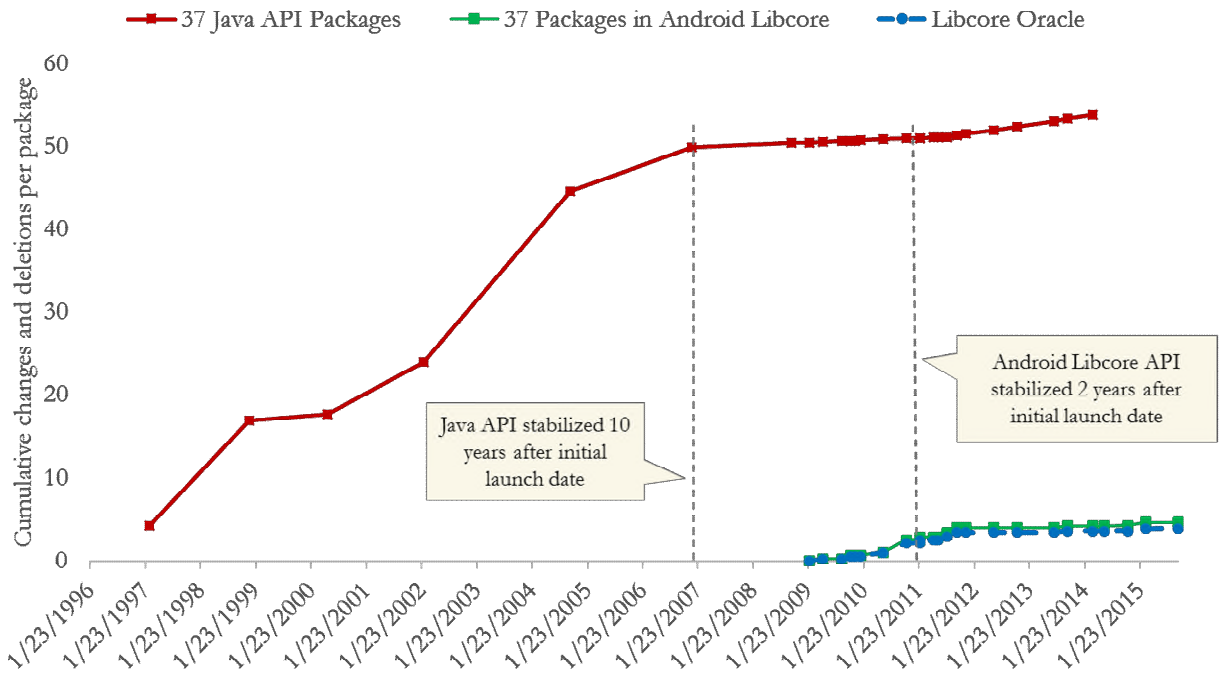
Figure 3: Cumulative Framework Changes and Deletions per Package (Google vs Oracle)



17. The same analysis was conducted for all Java APIs in the JDK, this time counting both changes and deletions. As illustrated in Figure 4, the Java APIs stabilized around 10.9 years after the first release of the JDK, and only after undergoing an average of almost 50 changes and deletions per package. In contrast, the Android Libcore APIs, which are driven by the Java APIs from the 37 copied Java API packages, stabilized around Android API level 11 in February, 2011, only 2.3 years after the first release of Android. Absent the copied Java APIs, it is reasonable to assume that the evolution of the Android Libcore APIs would have exhibited a trajectory more like both the early years of the Java APIs and the early years of the Google-written Framework APIs. It is also reasonable to assume that developers' adoption of Android APIs would have been slower without the copied Java APIs. This would be consistent with the finding of McDonnell *et al.* (2013) who concluded that "developers seem hesitant to embrace unstable, fast-evolving APIs quickly."[13]

[13] McDonnell, T., Ray, B. and Kim, M. (2013). "An Empirical Study of API Stability and Adoption in the Android Ecosystem." McDonnell *et al.* report that they are measuring changing using "apidiff", which shows changes in declaring code in public API declarations. They note that "changed methods and fields are the ones with a modified signature since the previous version". (pp. 3-4)

Figure 4: Copying as an 8-Year Market Head Start[14]



18. Thus, the results of the stability analysis remain the same regardless of whether deleted methods are included or not. Further, I conducted a sensitivity analysis, accounting for both changed methods and deleted methods, under each of Dr. Astrachan's asserted alternative scenarios addressed in my earlier report (excluding 61 classes and excluding the Java.lang package, respectively). When accounting for changed methods and deleted methods, under both conditions (excluding the 61 classes and excluding java.lang), the results of the stability analysis did not change. The points of relative stability reached in the Java APIs and Android APIs remained the same under each of these scenarios.

19. Dr. Astrachan also asserts that he does not understand what was counted as a method change in my initial analysis, and that it is not clear whether that means that there is one method change regardless of how many changes in a particular method occur, or whether each change in a particular method counts such that the method change number could be greater than one for that method.[15] Any change in a particular method's method structure means that method is counted as changed. As such, it is not possible for the method

[14] For the analysis of the JDK Java APIs, only publicly facing APIs are included since these are the only ones for which the number of changes can be measured. The following 5 APIs are excluded from use in this analysis: javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, and javax.net.ssl. While specified in the Java API specification (http://docs.oracle.com/javase/1.5.0/docs/api/javax/crypto/package-summary.html), these API packages are not included in the standard JDK 5.0 download (available at http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase5-419410.html) or earlier versions of the JDK. In order to use a consistent data set over time, these 5 APIs are excluded.

[15] Astrachan Rebuttal Report, ¶ 174.

change number to be greater than one for a given method between two subsequent versions. For clarification, another way to express the count is to refer to the "number of changed methods," which is the term adopted here to avoid any confusion.

20. Dr. Astrachan also takes issue with the graphing of the cumulative number of changes per package over time, rather than showing changes only within particular periods of time.[16] This criticism is unfounded, as the graph showing the cumulative number of changes per package serves *both* purposes: (1) it shows the trend over time, which is relevant here, and (2) the changes within any particular period of time are represented by the change in the slope of the line. A period with a steep slope represents a greater number of changes than a period with a flatter slope. Thus, the information Dr. Astrachan seeks is already present in the graph. However, in addition, we have also provided the actual data used in this analysis to Google.

21. Overall, graphical representations and a general sense of the relevance of cumulative behavior over time are common both in the research literature and are familiar in people's everyday experiences. For example, in the research literature on technology or new product adoption it is quite common to plot the cumulative degree of adoption, e.g., the percentage of a population over time that has adopted a new technology or product, generally shown as an S-shaped curve. Cumulative amounts over time are also common decision variables in everyday decision making. For example, in answer to the question, "Are you financially prepared for retirement?" the answer would rest, in part, on the cumulative amount of money that a person has saved over his or her lifetime. Or, in a negative vein, if a criminal were being sentenced, a judge might consider how many similar crimes that individual had committed over his or her lifetime, e.g. was this the first DUI conviction, or the third? Cumulative data are both useful and familiar.

22. Therefore, in a similar vein, the plot of cumulative changes here shows the evolution of the stability of the Android platform over time. This information helps to form the reputation of the platform to its audience, software developers. Further, recent research (as cited above, and in my earlier reports) shows that, all else being equal, developers prefer to write to a stable platform, and that lower quality apps are associated with APIs that are relatively unstable.

23. Dr. Astrachan suggests that the stability analysis presented in my prior reports does not show an actual contribution to the stability of Android.[17] This assertion ignores the data that was already analyzed in my prior reports. The same data regarding stability of Android APIs created by Google[18] and stability of the 37 copied Java API packages[19] represented graphically in my prior reports, can be expressed numerically as a

[16] Astrachan Rebuttal Report, ¶ 175.
[17] *See e.g.* Astrachan Rebuttal Report, ¶ 172
[18] Called "Libcore Google" in my prior reports. *See e.g.* Initial Report, Figure 5.
[19] Called "Libcore Oracle Infringed" in my prior reports. *See e.g.* Initial Report, Figure 5.

percentage increase in stability of the Android APIs from the presence of the Oracle APIs. This shows a direct contribution of the copied 37 Java API packages to the stability of Android APIs.

24. The objective of this analysis is to determine the amount of stability that the copied Java APIs from the 37 Java API packages contribute to Android Libcore. I consider an alternative, but-for scenario in which the 37 copied packages in Libcore would have been written by Google instead of being copied from Oracle. I make the assumption that, all else being equal, the 37 newly written Google APIs would exhibit the average change behavior of other Google-written APIs. By comparing this alternative scenario of 37 Google-written APIs to the actual case of the 37 copied Oracle APIs I can estimate the percentage increase in stability that Android Libcore achieves through its reliance on the 37 copied Oracle APIs.

25. The steps to derive this percentage are as follows:

(1) Sum up all changed methods across the entire version history to obtain two values: one for the total number of methods changed within the Libcore Google packages, and another for the 37 copied packages in Libcore.

(2) Count the number of common packages between the two most recent consecutive API Levels (levels 22 and 23) for the Libcore Google packages.

(3) Divide the number of methods changed in the Libcore Google packages (from step 1) by the number of common packages (from step 2) to yield the average number of methods changed per Libcore Google package across Android's entire version history.

(4) Multiply the result of step 3 (the number of average changes per Libcore Google package) by 37 in order to determine the total number of changes that one would expect to observe, all else being equal, if the 37 newly written packages exhibited the same level of stability as an average actual Google written package in Libcore.

(5) Compare the result of step 4 to the total number of changed methods actually observed in the 37 copied packages.

26. The results of this methodology are set forth in Table 1.

Table 1: Contribution of Oracle's 37 copied packages to the Stability of the Android Libcore

	Google-written Libcore	37 Copied Packages in Libcore
Total number of Methods Changed[20]	90	118
Number of Packages[21]	5	37
Average number of Methods Changed per Package	18.0	3.2
	But-for Estimate	**37 Copied Packages in Libcore**
Estimated Number of Methods Changed	18 * 37 = 666	
Actual number of Methods Changed		118
Estimated percentage stability improvement from copied Oracle APIs	(666-118)/666 = 82.3%	

27. Taking the average number of methods changed per Libcore Google package (18), and multiplying it across a set of 37 newly-written packages, reveals that these 37 packages could be expected to exhibit 666 changed methods over the course of Android's version history. Comparatively, the 118 methods changed in the 37 copied packages represent approximately an approximately 82% reduction from this result. By comparing the 37 copied packages to the but-for alternative scenario in which they would have been newly created by Google the significant contribution of the copied packages to the stability of Libcore is readily apparent.

28. As discussed in my initial report, Google itself understood that using Java would contribute to the stability of the Android platform and would be critical to its success. Google's own documents indicate that using Java as the basis for Android would decrease Android's time to market and reduce the risk of an unstable, buggy platform. For example, in 2006 Google observed: "We will ship a more stable product sooner if we do as much as possible in Java" [underlined emphasis added])[22] Later that same year Google employees again recognized that using Java was necessary to keep Android on schedule: "if the device is not fast and stable we FAIL" [underlined emphasis added]; "we are building a java based system: that decision is final"; "Java not Sun . . . we are building a java based system, not pushing Sun's agendas"; "Any significant change we make will disrupt the schedules."[23]

[20] This is across all Android versions.
[21] This is the number common to the two most recent Android versions.
[22] Initial Report, ¶ 81 (GOOGLE-01-00075935 at 935 (4/4/2006)).
[23] Initial Report, ¶ 83 (TX 23, GOOGLE-04-00055098 (8/16/2006)).

V. App Dependency Analysis

29. In Exhibit 2b of his report, Dr. Leonard provides his list of "top apps" from 2012 to 2013. In his report, Dr. Leonard asserts:

> "[a] substantial number of applications, including many of the most popular applications offered through the Google Play store, were in fact developed using the NDK. I assembled a list of apps that appeared in the daily top 100 download lists for both Android and iOS during the period 2012 to 2013. Google personnel identified whether each app was written in Java or C/C++. Of the apps on the list for which the language can be determined, 162 were written in C/C++."[24]

30. This paragraph seems to suggest that these apps do not use Java, and therefore do not rely on the 37 copied Oracle APIs. There is no empirical support for Dr. Leonard's assertion in this regard, beyond the claim that unidentified "Google personnel" purportedly identified whether apps were written in Java or C/C++. However, Dr. Leonard's analysis is not consistent with direct and objective analysis of the code that constitutes these apps.

31. As set forth in my earlier report, an analysis of the source code from the top 100 apps downloaded from the Google Play store reveals that every one of the top 100 apps depends upon a minimum of three of the 37 copied Java API packages.[25] The average number of dependencies is 11.5, or nearly a third of the 37 copied API packages. And, one of the top 100 apps depends on 23 of the 37 copied API packages.[26] If the analysis is restricted to the most popular of the 100 apps (the 14 apps that are listed as having between 1,000,000,000 and 5,000,000,000 downloads), they can be seen as being even more dependent upon the 37 copied API packages.[27] The minimum number of copied API packages these apps depend on is eight, the average number 13.8, and the maximum number 17.[28]

32. Further, I analyzed the dependencies on the 37 copied API packages in individual apps listed in Dr. Leonard's Exhibit 2b, and which were addressed in my initial report. This data is attached at Appendix B. This analysis also reveals the particular apps that Dr. Leonard asserts to be written only in C/C++ actually use the Java APIs from the 37 copied Java API packages. For example, Dr. Leonard asserts that the popular "Angry Birds Rio," "Fruit Ninja Free," "Candy Crush Saga," "Dropbox," "eBay," "Instagram," "Snapchat" and "Twitter" apps are purportedly written only in C/C++.[29] However, direct analysis of the source code of

[24] Leonard Report, ¶ 76, Exhibit 2b.
[25] Initial Report, ¶¶ 126-140.
[26] Initial Report, ¶¶ 126-140.
[27] Initial Report, ¶¶ 126-140.
[28] Initial Report, ¶¶ 126-140.
[29] Leonard Report, Exhibit 2b.

these apps shows that each of these apps uses and depends upon the Java APIs from the 37 copied Java API packages, meaning that they all must have at least some Java source code.[30]

Table 2: Dependence of Leonard-identified "NDK" Android apps on the 37 copied packages

App	Number of Copied Package Dependencies
Angry Birds Rio	8
Fruit Ninja Free	11
Candy Crush Saga	9
Dropbox	13
eBay	15
Instagram	15
Snapchat	14
Twitter	14

33. These apps each contain dependencies on a minimum of 8 of the 37 API packages, with the "eBay" and "Instagram" apps being dependent on 15 packages. Dr. Leonard's assertion that these apps are written in C/C++ is contradicted by the significant dependence they exhibit on the 37 copied Java API packages.

34. The foregoing list is illustrative only, and other examples can be seen through a comparison of Exhibit 2b of the Leonard Report to Appendix B of this report.

35. Similarly, Google asserts that its own "Google Earth," "Google Translate" and "YouTube" apps are written only in C/C++.[31] However, as established in my initial report, Google Earth uses and depends upon 12 of the 37 Java API packages, Google Translate uses and depends upon 14 of the 37 Java API packages and YouTube uses and depends upon 11 of the 37 Java API packages.[32]

36. Further, Dr. Leonard's assertion that the top apps listed in his report are not written in Java can be seen to be incorrect by counting the number of lines of Java code in these apps. An analysis of the lines of Java code in apps listed by Dr. Leonard in his Exhibit 2b as "NDK" apps was conducted as follows. Several of the apps classified as "NDK" apps were downloaded, and decompiled into their Java source files.

- Step 1: Extract the app's dex bytecode from the downloaded .apk file as one would for an ordinary .zip file.
- Step 2: Use the dex2jar tool to convert the dex bytecode into class bytecode.[33]
- Step 3: Use the CFR de-compiler tool to convert the class bytecode into Java source code.[34]

[30] Appendix B.
[31] Leonard Report, Exhibit 2b.
[32] Initial Report, Table 1.
[33] This step uses the dex2jar tool (see http://sourceforge.net/projects/dex2jar/)

- Step 4: Use the SlocCount tool to count the total number of lines of Java source code for each app.[35]

37. By way of example, I list here the number of lines of Java code in each of the highly popular apps listed above.

Table 3: Number of lines of Java code in Android "NDK" apps

App	Number of Lines of Java Code
Angry Birds Rio	428,593
Fruit Ninja Free	477,262
Candy Crush Saga	106,070
Dropbox	354,741
eBay	512,870
Instagram	427,656
Snapchat	849,800
Twitter	187,420

38. The above results clearly demonstrate the inconsistencies in Dr. Leonard's classification system. Specifically, they show that these apps all contain Java source code on the order of hundreds of thousands of lines of code, and yet are classified by Leonard as "NDK" apps rather than as "Java" apps. Dr. Leonard's assertions are inconsistent with the actual data. This direct analysis of the source code of the Google Play Store's most popular apps demonstrates that Dr. Leonard is incorrect in his assertions that top Android apps are written only in C/C++. Rather, those apps depend on the 37 Java API packages in a significant way. They also contain numerous lines of Java code, sometimes many more than apps that he classifies as being written in Java.[36] In Dr. Leonard's Exhibit 2b he indicates "Programming languages are identified by Google" for each of the listed apps. Dr. Leonard's analysis is not based on actual analysis of the source code for the applications listed in his Exhibit 2b, and therefore its conclusions cannot be relied on.

[34] This step uses a tool called CFR – another Java Decompiler (see http://www.benf.org/other/cfr/). This tool accepts the output of the Dex2Jar tool, and de-compiles it into the app's original Java source code. Several Java Decompilers were evaluated. Out of the available Open Source Java Decompilers, CFR was selected due to its ongoing development and support of modern Java 8 lambda features.

[35] The Sloccount tool (http://www.dwheeler.com/sloccount/) reads the de-compiled source files for a given app and counts the number of total lines.

[36] I also examined the "Bubble Mania" app, classified as a "Java" app in Dr. Leonard's exhibit. This app was found to contain roughly 7,000 lines of Java code. The "Candy Crush Saga" app is classified as an "NDK" app and contains more than 15 times as many lines of Java code, at over 106,000.

TABLE OF APPENDICES

APPENDIX A – Materials Considered

A. Court Documents

- Expert Rebuttal Report of Dr. Owen Astrachan
- Expert Report of Dr. Adam Jaffe
- Expert Report of Dr. Gregory K. Leonard
- GOOGLE-01-00075935
- GOOGLE-04-00055098
- Initial Expert Report of Dr. Chris F. Kemerer
- TX 23
- Deposition of Andrew E. Rubin, July 27, 2011, p. 180.

B. Public Sources

- *CFR- Another Java Decompiler,* BenF, http://www.benf.org/other/cfr (last visited February 28, 2016).
- *Dex2jar,* SourceForge (Mar. 28, 2015), http://sourceforge.net/projects/dex2jar.
- *Java SE 5.0 Downloads,* Oracle, http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase5-419410.html (last visited February 28, 2016).
- *Package javax.crypto,* Oracle, http://docs.oracle.com/javase/1.5.0/docs/api/javax/crypto/package-summary.html (last visited February 28, 2016).
- *SLOCCount,* D Wheeler, http://www.dwheeler.com/sloccount/ (last visited February 28, 2016).
- Tyler McDonnell, Ray Baishakhi, Kim Miryung, "An Empirical Study of API Stability and Adoption in the Android Ecosystem", UCLA (2013), http://web.cs.ucla.edu/~miryung/Publications/icsm2013-apiecosystem.pdf.
- Mario Linares-Vásquez, Gabriele Bavota, Carlos Bernal-Cárdenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2013. "API change and fault proneness: a threat to the success of Android apps". *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering* (ESEC/FSE 2013). ACM, New York, NY, USA, 477-487.
- Yuan Tian, M. Nagappan, D. Lo and A. E. Hassan, "What are the characteristics of high-rated apps? A case study on free Android Applications," *2015 IEEE International Conference on Software Maintenance and Evolution* (ICSME), , Bremen, 2015, pp. 301-310.

APPENDIX B – Dependencies on 37 APIs by top Android Apps

App Name	java.net	java.nio	java.sql	java.lang.annotation	java.lang.reflect	java.util.logging	javax.net	javax.net.ssl	javax.security.auth.login	javax.sound.midi	java.nio.channels	java.awt.image	javax.security	javax.text	javax.security.cert	javax.crypto.interfaces	java.beans	java.security.interfaces	javax.security.auth.callback
Facebook	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Messenger	1	1	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0
Google Play Books	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Drive	1	1	0	0	1	1	0	1	0	0	0	1	1	1	0	0	0	0	0
Google Play Newsstand	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Maps	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Gmail	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Talkback	1	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Google Play Music	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Play Games	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Text-to-speech	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	0	0
Google Play Movies & TV	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
WhatsApp Messenger	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Clean Master (Boost & AppLock)	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Google Street View	1	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
Instagram	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Samsung Push Service	1	1	0	0	1	0	1	0	1	0	0	1	1	1	1	0	0	0	1
Skype - free IM & video calls	1	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0	0	0	0
Cymera - Photo Editor, Collage	1	1	0	1	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Tiny Flashlight + LED	1	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
DU Speed	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0

App Name	java.net	java.nio	java.sql	java.lang.annotation	java.lang.reflect	java.util.logging	javax.net	javax.lang.ref	javax.security.auth.login	javax.net.ssl	java.nio.channels	java.nio	javax.security	javax.text	javax.security.cert	javax.crypto.interfaces	java.beans	java.security.interfaces	javax.security.auth.callback
Booster Cache Cleaner																			
AppLock	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Dropbox	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
ANT+ Plugins Service	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
ANT Radio Service	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
Real Racing 3	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0
eBay	1	0	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
ES File Explorer File Manager	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Evernote	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Hill Climb Racing	1	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
Pool Billiards Pro	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
Despicable Me	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
GO Launcher -Theme & Wallpaper	1	1	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Cloud Print	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	0	0
Google News & Weather	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Translate	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Calendar	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Google Keyboard	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Google Earth	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Fruit Ninja Free	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Jetpack Joyride	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
HP Print Service Plugin	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Battery Doctor (Battery Saver)	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Temple Run 2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
imo free video calls and chat	1	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0

App Name	java.net	java.nio	java.sql	java.lang.annotation	java.lang.reflect	java.util.logging	javax.net	javax.security.auth.login	javax.net.ssl	java.nio.channels	java.nio	javax.security	javax.text	javax.security.cert	javax.crypto.interfaces	java.beans	java.security.interfaces	javax.security.auth.callback
GO SMS Pro	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Candy Crush Saga	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Candy Crush Soda Saga	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Farm Heroes Saga	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Pet Rescue Saga	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Omni Swipe	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
Lookout Security & Antivirus	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
8 Ball Pool	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
MX Player	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
Glow Hockey	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Netflix	1	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
My Talking Angela	1	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
My Talking Tom	1	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Talking Ginger	1	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0
Talking Tom Cat 2	1	1	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
PandoraÂ® Radio	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
PicsArt Photo Studio	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
360 Security - Antivirus Boost	1	1	1	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Photo Grid - Collage Maker	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Angry Birds Rio	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Angry Birds Seasons	1	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
S Health - Fitness Diet Tracker	1	1	0	0	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1
Samsung Print Service	1	1	0	0	1	1	0	1	1	0	0	1	1	1	1	0	0	0	1

App Name	java.net	java.nio	java.sql	java.lang.annotation	java.lang.reflect	java.util.logging	javax.net	javax.lang.ref	javax.security.auth.login	javax.net.ssl	java.nio.channels	java.nio	javax.security	javax.text	javax.security.cert	javax.crypto.interfaces	java.beans	java.security.interfaces	javax.security.auth.callback
Plugin																			
Tango - Free Video Call & Chat	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Shazam	1	1	0	1	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Snapchat	1	1	0	1	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Smart Connect	1	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Spotify Music	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Clash of Clans	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0
Hay Day	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0
4shared	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
TripAdvisor Hotels Flights	1	1	1	1	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Twitter	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
UC Browser - Fast Download	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
Don't Tap The White Tile	1	0	1	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Retrica	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Viber	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Yahoo Mail â€“ Free Email App	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
Cut the Rope FULL FREE	1	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Flipboard: Your News Magazine	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	0	0
Kik	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Pou	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
ZEDGEâ„¢ Ringtones & Wallpapers	1	1	0	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0
Firefox Browser for Android	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
TuneIn Radio - Radio & Music	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0

App Name	java.net	java.nio	java.sql	java.lang.annotation	java.lang.reflect	java.util.logging	javax.net	javax.lang.ref	javax.security.auth.login	javax.net.ssl	java.nio.channels	javax.io	javax.security	javax.text	javax.security.cert	javax.crypto.interfaces	javax.beans	java.security.interfaces	javax.security.auth.callback
Peel Smart Remote (WatchONâ„¢)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Camera360 Ultimate	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Kik	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
Pou	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
ZEDGEâ„¢ Ringtones & Wallpapers	1	1	0	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0
Firefox Browser for Android	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0
TuneIn Radio - Radio & Music	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0
Peel Smart Remote (WatchONâ„¢)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Camera360 Ultimate	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Total	**94**	**77**	**15**	**17**	**90**	**27**	**43**	**76**	**3**	**0**	**0**	**98**	**86**	**77**	**5**	**0**	**1**	**0**	**3**

App Name	java.util.jar	java.awt.font	java.util.zip	java.util.regex	java.lang	java.util.prefs	java.util	java.security.cert	java.nio.channels.spi	java.security.spec	javax.crypto	java.nio.charset.spi	javax.crypto.spec	javax.sql	java.nio.charset	java.security.acl	javax.security.auth.x500	javax.security.auth
Facebook	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
Messenger	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0
Google Play Books	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Google Drive	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Play Newsstand	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
Maps	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Gmail	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Google	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Google Talkback	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Play Music	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
Google Play Games	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Text-to-speech	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Google Play Movies & TV	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
WhatsApp Messenger	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Clean Master (Boost & AppLock)	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Google Street View	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Instagram	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Samsung Push Service	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	1
Skype - free IM & video calls	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Cymera - Photo Editor, Collage	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0
Tiny Flashlight + LED	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
DU Speed Booster Cache Cleaner	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

App Name	java.util.jar	java.awt.font	java.util.zip	java.util.regex	java.lang	java.util.prefs	java.util	java.security.cert	java.nio.channels.spi	java.security.spec	javax.crypto	java.nio.charset.spi	javax.crypto.spec	javax.sql	java.nio.charset	java.security.acl	javax.security.auth.x500	javax.security.auth
AppLock	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Dropbox	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
ANT+ Plugins Service	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
ANT Radio Service	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Real Racing 3	1	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	1	1
eBay	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
ES File Explorer File Manager	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Evernote	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Hill Climb Racing	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Pool Billiards Pro	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Despicable Me	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
GO Launcher -Theme & Wallpaper	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Cloud Print	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google News & Weather	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Translate	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Calendar	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Keyboard	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Google Earth	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Fruit Ninja Free	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Jetpack Joyride	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
HP Print Service Plugin	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Battery Doctor (Battery Saver)	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Temple Run 2	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
imo free video calls and chat	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
GO SMS Pro	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Candy Crush Saga	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Candy	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0

App Name	java.util.jar	java.awt.font	java.util.zip	java.util.regex	java.lang	java.util.prefs	java.util	java.security.cert	java.nio.channels.spi	java.security.spec	javax.crypto	java.nio.charset.spi	javax.crypto.spec	javax.sql	java.nio.charset	java.security.acl	javax.security.auth.x500	javax.security.auth
Crush Soda Saga																		
Farm Heroes Saga	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Pet Rescue Saga	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Omni Swipe	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Lookout Security & Antivirus	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
8 Ball Pool	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
MX Player	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Glow Hockey	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Netflix	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
My Talking Angela	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
My Talking Tom	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Talking Ginger	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Talking Tom Cat 2	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
PandoraÂ® Radio	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
PicsArt Photo Studio	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
360 Security - Antivirus Boost	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Photo Grid - Collage Maker	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Angry Birds Rio	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Angry Birds Seasons	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
S Health - Fitness Diet Tracker	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	1
Samsung Print Service Plugin	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	1
Tango - Free Video Call & Chat	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Shazam	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Snapchat	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0

App Name	java.util.jar	java.awt.font	java.util.zip	java.util.regex	java.lang	java.util.prefs	java.util	java.security.cert	java.nio.channels.spi	java.security.spec	javax.crypto	java.nio.charset.spi	javax.crypto.spec	javax.sql	java.nio.charset	java.security.acl	javax.security.auth.x500	javax.security.auth
Smart Connect	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Spotify Music	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Clash of Clans	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Hay Day	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
4shared	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
TripAdvisor Hotels Flights	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Twitter	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
UC Browser - Fast Download	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Don't Tap The White Tile	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Retrica	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Viber	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Yahoo Mail â€“ Free Email App	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Cut the Rope FULL FREE	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Flipboard: Your News Magazine	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Kik	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Pou	1	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	1	0
ZEDGEâ„¢ Ringtones & Wallpapers	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
Firefox Browser for Android	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
TuneIn Radio - Radio & Music	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Peel Smart Remote (WatchONâ„¢)	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Camera360 Ultimate	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Kik	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
Pou	1	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	1	0

App Name	java.util.jar	java.awt.font	java.util.zip	java.util.regex	java.lang	java.util.prefs	java.util	java.security.cert	java.nio.channels.spi	java.security.spec	javax.crypto	java.nio.charset.spi	javax.crypto.spec	javax.sql	java.nio.charset	java.security.acl	javax.security.auth.x500	javax.security.auth
ZEDGEâ„¢ Ringtones & Wallpapers	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
Firefox Browser for Android	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0
TuneIn Radio - Radio & Music	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Peel Smart Remote (WatchONâ„¢)	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Camera360 Ultimate	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Total	**15**	**3**	**62**	**84**	**1000**	**4**	**1000**	**0**	**0**	**0**	**52**	**0**	**0**	**3**	**0**	**0**	**16**	**4**

APPENDIX C – R Script for Calculating Package Change across Java SE Versions

```
install.packages("dplyr")
library(dplyr)

m_unique <- java_stability[java_stability$count==1,]
m_multiple <- java_stability[java_stability$count > 1,]

#m_multiple$m_str_var <- paste(m_multiple$m_struct,m_multiple$m_var)

#write.csv(m_multiple,file='m_multiple.csv')

m_multiple_group <- group_by(m_multiple,class,package,method,version)
m_multiple_updated <- summarize(m_multiple_group,m_struct_c =
paste(m_struct, collapse = " "))

#android_stability$m_str_var <-
paste(android_stability$m_struct,android_stability$m_var)
#android_stability$m_char <- nchar(android_stability$m_str_var)


p <- c("java.applet",
       "java.awt",
       "java.awt.image",
       "java.awt.peer",
       "java.io",
       "java.lang",
       "java.net",
       "java.util",
       "java.awt.datatransfer",
       "java.awt.event",
       "java.beans",
       "java.lang.reflect",
       "java.math",
       "java.rmi",
       "java.rmi.dgc",
       "java.rmi.registry",
       "java.rmi.server",
       "java.security",
       "java.security.acl",
       "java.security.interfaces",
       "java.sql",
       "java.text",
       "java.text.resources",
       "java.util.zip",
       "java.awt.color",
       "java.awt.dnd",
       "java.awt.dnd.peer",
       "java.awt.font",
       "java.awt.geom",
       "java.awt.im",
       "java.awt.image.renderable",
       "java.awt.print",
       "java.beans.beancontext",
       "java.lang.ref",
       "java.rmi.activation",
       "java.security.cert",
```

```
        "java.security.spec",
        "java.util.jar",
        "javax.accessibility",
        "javax.swing",
        "javax.swing.border",
        "javax.swing.colorchooser",
        "javax.swing.event",
        "javax.swing.filechooser",
        "javax.swing.plaf",
        "javax.swing.plaf.basic",
        "javax.swing.plaf.metal",
        "javax.swing.plaf.multi",
        "javax.swing.table",
        "javax.swing.text",
        "javax.swing.text.html",
        "javax.swing.text.html.parser",
        "javax.swing.text.rtf",
        "javax.swing.tree",
        "javax.swing.undo",
        "org.omg.CORBA",
        "org.omg.CORBA.DynAnyPackage",
        "org.omg.CORBA.ORBPackage",
        "org.omg.CORBA.portable",
        "org.omg.CORBA.TypeCodePackage",
        "org.omg.CosNaming",
        "org.omg.CosNaming.NamingContextPackage",
        "java.awt.im.spi",
        "javax.naming",
        "javax.naming.directory",
        "javax.naming.event",
        "javax.naming.ldap",
        "javax.naming.spi",
        "org.omg.CORBA_2_3",
        "org.omg.CORBA_2_3.portable",
        "org.omg.SendingContext",
        "org.omg.stub.java.rmi",
        "java.nio",
        "java.nio.channels",
        "java.nio.channels.spi",
        "java.nio.charset",
        "java.nio.charset.spi",
        "java.util.logging",
        "java.util.prefs",
        "java.util.regex",
        "javax.imageio",
        "javax.imageio.event",
        "javax.imageio.metadata",
        "javax.imageio.plugins.jpeg",
        "javax.imageio.spi",
        "javax.imageio.stream",
        "javax.print",
        "javax.print.attribute",
        "javax.print.attribute.standard",
        "javax.print.event",
        "javax.rmi",
        "javax.rmi.CORBA",
        "javax.security.auth",
```

```
111         "javax.security.auth.callback",
112         "javax.security.auth.kerberos",
113         "javax.security.auth.login",
114         "javax.security.auth.spi",
115         "javax.security.auth.x500",
116         "javax.sql",
117         "javax.xml.parsers",
118         "javax.xml.transform",
119         "javax.xml.transform.dom",
120         "javax.xml.transform.sax",
121         "javax.xml.transform.stream",
122         "org.apache.crimson.jaxp",
123         "org.apache.crimson.parser",
124         "org.apache.crimson.tree",
125         "org.apache.crimson.util",
126         "org.apache.xalan.client",
127         "org.apache.xalan.extensions",
128         "org.apache.xalan.lib",
129         "org.apache.xalan.lib.sql",
130         "org.apache.xalan.processor",
131         "org.apache.xalan.res",
132         "org.apache.xalan.serialize",
133         "org.apache.xalan.templates",
134         "org.apache.xalan.trace",
135         "org.apache.xalan.transformer",
136         "org.apache.xalan.xslt",
137         "org.apache.xml.dtm",
138         "org.apache.xml.dtm.ref",
139         "org.apache.xml.dtm.ref.dom2dtm",
140         "org.apache.xml.dtm.ref.sax2dtm",
141         "org.apache.xml.utils",
142         "org.apache.xml.utils.res",
143         "org.apache.xml.utils.synthetic",
144         "org.apache.xml.utils.synthetic.reflection",
145         "org.apache.xpath",
146         "org.apache.xpath.axes",
147         "org.apache.xpath.compiler",
148         "org.apache.xpath.functions",
149         "org.apache.xpath.objects",
150         "org.apache.xpath.operations",
151         "org.apache.xpath.patterns",
152         "org.apache.xpath.res",
153         "org.ietf.jgss",
154         "org.omg.CosNaming.NamingContextExtPackage",
155         "org.omg.Dynamic",
156         "org.omg.DynamicAny",
157         "org.omg.DynamicAny.DynAnyFactoryPackage",
158         "org.omg.DynamicAny.DynAnyPackage",
159         "org.omg.IOP",
160         "org.omg.IOP.CodecFactoryPackage",
161         "org.omg.IOP.CodecPackage",
162         "org.omg.Messaging",
163         "org.omg.PortableInterceptor",
164         "org.omg.PortableInterceptor.ORBInitInfoPackage",
165         "org.omg.PortableServer",
166         "org.omg.PortableServer.CurrentPackage",
167         "org.omg.PortableServer.POAManagerPackage",
```

```
168         "org.omg.PortableServer.POAPackage",
169         "org.omg.PortableServer.portable",
170         "org.omg.PortableServer.ServantLocatorPackage",
171         "org.w3c.dom",
172         "org.w3c.dom.css",
173         "org.w3c.dom.events",
174         "org.w3c.dom.html",
175         "org.w3c.dom.stylesheets",
176         "org.w3c.dom.traversal",
177         "org.w3c.dom.views",
178         "org.xml.sax",
179         "org.xml.sax.ext",
180         "org.xml.sax.helpers",
181         "java.lang.annotation",
182         "java.lang.instrument",
183         "java.lang.management",
184         "java.util.concurrent",
185         "java.util.concurrent.atomic",
186         "java.util.concurrent.locks",
187         "javax.imageio.plugins.bmp",
188         "javax.management",
189         "javax.management.loading",
190         "javax.management.modelmbean",
191         "javax.management.monitor",
192         "javax.management.openmbean",
193         "javax.management.relation",
194         "javax.management.remote",
195         "javax.management.remote.rmi",
196         "javax.management.timer",
197         "javax.rmi.ssl",
198         "javax.security.sasl",
199         "javax.sound.midi",
200         "javax.sound.midi.spi",
201         "javax.sound.sampled",
202         "javax.sound.sampled.spi",
203         "javax.sql.rowset",
204         "javax.sql.rowset.serial",
205         "javax.sql.rowset.spi",
206         "javax.swing.plaf.synth",
207         "javax.xml",
208         "javax.xml.datatype",
209         "javax.xml.namespace",
210         "javax.xml.validation",
211         "javax.xml.xpath",
212         "org.w3c.dom.bootstrap",
213         "org.w3c.dom.ls",
214         "org.w3c.dom.ranges",
215         "java.text.spi",
216         "java.util.spi",
217         "javax.annotation",
218         "javax.annotation.processing",
219         "javax.lang.model",
220         "javax.lang.model.element",
221         "javax.lang.model.type",
222         "javax.lang.model.util",
223         "programelements",
224         "types",
```

```
225         "javax.script",
226         "javax.tools",
227         "javax.xml.bind",
228         "javax.xml.bind.annotation",
229         "javax.xml.bind.annotation.adapters",
230         "javax.xml.bind.attachment",
231         "javax.xml.bind.helpers",
232         "javax.xml.bind.util",
233         "javax.xml.crypto",
234         "javax.xml.crypto.dom",
235         "javax.xml.crypto.dsig",
236         "javax.xml.crypto.dsig.dom",
237         "javax.xml.crypto.dsig.keyinfo",
238         "javax.xml.crypto.dsig.spec",
239         "javax.xml.soap",
240         "javax.xml.stream",
241         "javax.xml.stream.events",
242         "javax.xml.stream.util",
243         "javax.xml.transform.stax",
244         "javax.xml.ws",
245         "javax.xml.ws.handler",
246         "javax.xml.ws.handler.soap",
247         "javax.xml.ws.http",
248         "javax.xml.ws.soap",
249         "javax.xml.ws.spi",
250         "org.w3c.dom.xpath",
251         "java.lang.invoke",
252         "java.nio.file",
253         "java.nio.file.attribute",
254         "java.nio.file.spi",
255         "javax.security.cert",
256         "javax.swing.plaf.nimbus",
257         "javax.xml.ws.spi.http",
258         "javax.xml.ws.wsaddressing",
259         "java.time",
260         "java.time.chrono",
261         "java.time.format",
262         "java.time.temporal",
263         "java.time.zone",
264         "java.util.function",
265         "java.util.stream")
266
267 # method change analysis
268
269 change_unique <- data.frame(matrix(ncol = 8, nrow = 248))
270 change_multiple <- data.frame(matrix(ncol = 8, nrow = 248))
271 change <- data.frame(matrix(ncol = 8, nrow = 248))
272
273 for (j in 1:248)
274 {
275   p_unique <- m_unique[m_unique$package == p[j],]
276   p_multiple <- m_multiple_updated[m_multiple_updated$package == p[j],]
277
278   for (i in 1:8)
279   {
280     p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
```

```
281     p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
282     change_unique[j,i] <- nrow(p_change_unique[p_change_unique$change ==
FALSE,])
283
284     p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y =
p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
285     p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
286     change_multiple[j,i] <-
nrow(p_change_multiple[p_change_multiple$change == FALSE,])
287
288     change <- change_unique + change_multiple
289
290   }
291
292 }
293
294 write.csv(change,file='change.csv')
295
296
297 #method added and removed
298
299 java_maturity <- distinct(select(java_stability,package, class, method,
version))
300
301 method_add <- data.frame(matrix(ncol = 8, nrow = 248))
302 method_remove <- data.frame(matrix(ncol = 8, nrow = 248))
303
304 for (j in 1:248)
305 {
306   p_analysis <- java_maturity[java_maturity$package == p[j],]
307
308   for (i in 1:8)
309   {
310     p_merge <- merge(x = p_analysis[p_analysis$version==i,],y =
p_analysis[p_analysis$version==i+1,], by = c("package","class","method"),all
= TRUE)
311     method_add[j,i] <- nrow(p_merge[is.na(p_merge$version.x),])
312     method_remove[j,i] <- nrow(p_merge[is.na(p_merge$version.y),])
313
314   }
315
316 }
317
318 write.csv(method_add,file='method_add.csv')
319 write.csv(method_remove,file='method_remove.csv')
320
321 #write.csv(android_maturity,file = 'android_maturity.csv')
322
323 # method size for all API levels
324
325 method_size <- data.frame(matrix(ncol = 9, nrow = 248))
326
327 for (j in 1:248)
328 {
329   p_size <- java_maturity[java_maturity$package == p[j],]
```

```
330
331   for (i in 1:9)
332   {
333     method_size[j,i] <- nrow(p_size[p_size$version==i,])
334
335   }
336
337 }
338
339 write.csv(method_size,file='method_size.csv')
340
341 # 37 API size over versions
342
343 p_copied <- c("javax.sql",
344               "java.beans",
345               "java.lang.ref",
346               "java.net",
347               "java.util.logging",
348               "java.util",
349               "java.io",
350               "java.lang",
351               "java.lang.annotation",
352               "java.nio",
353               "java.nio.channels",
354               "java.nio.channels.spi",
355               "java.nio.charset",
356               "java.security",
357               "java.security.acl",
358               "java.security.cert",
359               "java.security.interfaces",
360               "java.sql",
361               "java.text",
362               "java.util.jar",
363               "java.util.prefs",
364               "java.util.zip",
365               "javax.crypto",
366               "javax.crypto.interfaces",
367               "javax.crypto.spec",
368               "javax.net",
369               "javax.security.auth",
370               "javax.security.auth.callback",
371               "javax.security.auth.login",
372               "javax.security.cert",
373               "java.nio.charset.spi",
374               "java.security.spec",
375               "java.util.regex",
376               "javax.net.ssl",
377               "javax.security.auth.x500",
378               "java.lang.reflect",
379               "java.awt.font")
```

APPENDIX D – R Script for Calculating Package Change across Android Versions

```
install.packages("dplyr")
library(dplyr)

#pull in the data files
setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/34_API stability analysis/android stability")

android_stability <- read.csv("android stability input.csv", as.is = TRUE)

#group the unique methods in each version and get the unique method list
android_stability <- android_stability %>% group_by(class,package,method,version) %>% mutate(length(m_struct))
names(android_stability)[8] <- "count"

#get the methods with only 1 method strucure in each version
m_unique <- android_stability[android_stability$count==1,]

#get the method with more than 1 method structure in each version
m_multiple <- android_stability[android_stability$count > 1,]
m_multiple$m_str_var <- paste(m_multiple$m_struct,m_multiple$m_var)

write.csv(m_multiple,file='m_multiple.csv')

m_multiple_group <- group_by(m_multiple,class,package,method,version)
m_multiple_updated <- summarize(m_multiple_group,m_struct_c = paste(m_struct, collapse = " "))


#android_stability$m_str_var <- paste(android_stability$m_struct,android_stability$m_var)
#android_stability$m_char <- nchar(android_stability$m_str_var)


p <- c("android",
       "android.annotation",
       "android.content.res",
       "java.nio.channels",
       "java.nio.channels.spi",
       "android.database",
       "java.nio.charset",
       "java.nio.charset.spi",
       "java.security",
       "java.security.acl",
       "java.security.cert",
       "java.security.spec",
       "java.sql",
       "java.text",
       "java.util",
       "android.database.sqlite",
       "java.util.concurrent",
       "java.util.concurrent.atomic",
       "java.util.concurrent.locks",
       "java.util.jar",
       "java.util.logging",
```

```
        "java.util.prefs",
        "java.util.regex",
        "java.util.zip",
        "javax.crypto",
        "javax.crypto.spec",
        "android.graphics",
        "javax.microedition.khronos.egl",
        "javax.net",
        "javax.net.ssl",
        "javax.security.auth",
        "javax.security.auth.callback",
        "javax.security.auth.login",
        "javax.security.auth.x500",
        "javax.security.cert",
        "javax.sql",
        "javax.xml",
        "javax.xml.parsers",
        "junit.framework",
        "junit.runner",
        "org.apache.http",
        "org.apache.http.auth",
        "org.apache.http.auth.params",
        "org.apache.http.client",
        "org.apache.http.client.entity",
        "org.apache.http.client.methods",
        "org.apache.http.client.params",
        "org.apache.http.client.protocol",
        "org.apache.http.client.utils",
        "org.apache.http.conn",
        "org.apache.http.conn.params",
        "org.apache.http.conn.routing",
        "org.apache.http.conn.scheme",
        "org.apache.http.conn.ssl",
        "org.apache.http.conn.util",
        "org.apache.http.cookie",
        "org.apache.http.cookie.params",
        "org.apache.http.entity",
        "org.apache.http.impl",
        "org.apache.http.impl.auth",
        "org.apache.http.impl.client",
        "org.apache.http.impl.conn",
        "org.apache.http.impl.conn.tsccm",
        "org.apache.http.impl.cookie",
        "org.apache.http.impl.entity",
        "org.apache.http.impl.io",
        "org.apache.http.message",
        "org.apache.http.params",
        "android.app",
        "org.apache.http.protocol",
        "org.apache.http.util",
        "org.json",
        "org.w3c.dom",
        "org.xml.sax",
        "org.xml.sax.ext",
        "org.xml.sax.helpers",
        "org.xmlpull.v1",
        "org.xmlpull.v1.sax2",
```

```
108         "android.graphics.drawable",
109         "android.graphics.drawable.shapes",
110         "android.hardware",
111         "android.location",
112         "android.media",
113         "android.net",
114         "android.net.http",
115         "android.net.wifi",
116         "android.opengl",
117         "android.os",
118         "android.preference",
119         "android.provider",
120         "android.sax",
121         "android.telephony",
122         "android.telephony.gsm",
123         "android.test",
124         "android.test.mock",
125         "android.test.suitebuilder",
126         "android.test.suitebuilder.annotation",
127         "android.text",
128         "android.text.method",
129         "android.text.style",
130         "android.content",
131         "android.text.util",
132         "android.util",
133         "android.view",
134         "android.view.animation",
135         "android.webkit",
136         "android.widget",
137         "dalvik.annotation",
138         "dalvik.system",
139         "java.awt.font",
140         "java.io",
141         "android.content.pm",
142         "java.lang",
143         "java.lang.annotation",
144         "java.lang.ref",
145         "java.lang.reflect",
146         "java.math",
147         "java.net",
148         "java.nio",
149         "com.android.internal.util",
150         "dalvik.bytecode",
151         "java.security.interfaces",
152         "javax.crypto.interfaces",
153         "javax.microedition.khronos.opengles",
154         "org.apache.commons.logging",
155         "org.apache.http.io",
156         "android.inputmethodservice",
157         "android.speech",
158         "android.text.format",
159         "android.appwidget",
160         "android.view.inputmethod",
161         "java.beans",
162         "android.gesture",
163         "android.accessibilityservice",
164         "android.speech.tts",
```

```
        "android.view.accessibility",
        "android.accounts",
        "android.telephony.cdma",
        "android.bluetooth",
        "android.service.wallpaper",
        "javax.xml.datatype",
        "javax.xml.namespace",
        "javax.xml.transform",
        "javax.xml.transform.dom",
        "javax.xml.transform.sax",
        "javax.xml.transform.stream",
        "javax.xml.validation",
        "javax.xml.xpath",
        "org.w3c.dom.ls",
        "android.app.admin",
        "android.app.backup",
        "android.media.audiofx",
        "android.net.sip",
        "android.nfc",
        "android.nfc.tech",
        "android.os.storage",
        "android.drm",
        "android.animation",
        "android.renderscript",
        "android.hardware.usb",
        "android.mtp",
        "android.net.rtp",
        "android.media.effect",
        "android.net.wifi.p2p",
        "android.security",
        "android.service.textservice",
        "android.view.textservice",
        "android.hardware.input",
        "android.net.nsd",
        "android.net.wifi.p2p.nsd",
        "android.hardware.display",
        "android.service.dreams",
        "android.hardware.location",
        "android.service.notification",
        "android.graphics.pdf",
        "android.nfc.cardemulation",
        "android.print",
        "android.print.pdf",
        "android.printservice",
        "android.transition",
        "android.app.job",
        "android.app.usage",
        "android.bluetooth.le",
        "android.hardware.camera2",
        "android.hardware.camera2.params",
        "android.media.browse",
        "android.media.projection",
        "android.media.session",
        "android.media.tv",
        "android.service.media",
        "android.service.restrictions",
        "android.service.voice",
```

```
222         "android.system",
223         "android.telecom",
224         "android.service.carrier",
225         "android.app.assist",
226         "android.hardware.fingerprint",
227         "android.media.midi",
228         "android.security.keystore",
229         "android.service.chooser")
230
231 # method change analysis
232
233 change_unique <- data.frame(matrix(ncol = 22, nrow = 200))
234 change_multiple <- data.frame(matrix(ncol = 22, nrow = 200))
235 change <- data.frame(matrix(ncol = 22, nrow = 200))
236
237 for (j in 1:200)
238 {
239   p_unique <- m_unique[m_unique$package == p[j],]
240   p_multiple <- m_multiple_updated[m_multiple_updated$package == p[j],]
241
242   for (i in 1:22)
243   {
244       p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
245       p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
246       change_unique[j,i] <- nrow(p_change_unique[p_change_unique$change
== FALSE,])
247
248       p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y
= p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
249       p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
250       change_multiple[j,i] <-
nrow(p_change_multiple[p_change_multiple$change == FALSE,])
251
252       change <- change_unique + change_multiple
253
254   }
255
256 }
257
258 write.csv(change,file='change.csv')
259
260 # method change for API level 13-14
261
262 android_1314 <- read.csv("13_14_input.csv", as.is = TRUE)
263 android_1314 <- android_1314 %>% group_by(class,package,method,version)
%>% mutate(length(m_struct))
264 names(android_1314)[7] <- "count"
265
266 m_unique_1314 <- android_1314[android_1314$count==1,]
267 m_multiple_1314 <- android_1314[android_1314$count > 1,]
268
269 m_multiple_1314group <-
group_by(m_multiple_1314,class,package,method,version)
```

```
m_multiple_1314updated <- summarize(m_multiple_1314group,m_struct_c =
paste(m_struct, collapse = " "))

change_unique1314 <- data.frame(matrix(ncol = 1, nrow = 200))
change_multiple1314 <- data.frame(matrix(ncol = 1, nrow = 200))
change1314 <- data.frame(matrix(ncol = 1, nrow = 200))

for (j in 1:200)
{
  p_unique1314 <- m_unique_1314[m_unique_1314$package == p[j],]
  p_multiple1314 <- m_multiple_1314updated[m_multiple_1314updated$package
== p[j],]

  p_change_unique1314 <- merge(x =
p_unique1314[p_unique1314$version==13,],y =
p_unique1314[p_unique1314$version==14,], by = c("class","method"),all =
FALSE)
  p_change_unique1314$change <- p_change_unique1314$m_struct.x ==
p_change_unique1314$m_struct.y
  change_unique1314[j,1] <-
nrow(p_change_unique1314[p_change_unique1314$change == FALSE,])

  p_change_multiple1314 <- merge(x =
p_multiple1314[p_multiple1314$version==13,],y =
p_multiple1314[p_multiple1314$version==14,], by = c("class","method"),all =
FALSE)
  p_change_multiple1314$change <- p_change_multiple1314$m_struct_c.x ==
p_change_multiple1314$m_struct_c.y
  change_multiple1314[j,1] <-
nrow(p_change_multiple1314[p_change_multiple1314$change == FALSE,])

  change1314 <- change_unique1314 + change_multiple1314

}

write.csv(change1314,file='change13_14.csv')


#method added and removed

android_maturity <- distinct(select(android_stability,package, class,
method, version))

method_add <- data.frame(matrix(ncol = 22, nrow = 200))
method_remove <- data.frame(matrix(ncol = 22, nrow = 200))

for (j in 1:200)
{
  p_analysis <- android_maturity[android_maturity$package == p[j],]

  for (i in 1:22)
  {
    p_merge <- merge(x = p_analysis[p_analysis$version==i,],y =
p_analysis[p_analysis$version==i+1,], by = c("package","class","method"),all
= TRUE)
    method_add[j,i] <- nrow(p_merge[is.na(p_merge$version.x),])
    method_remove[j,i] <- nrow(p_merge[is.na(p_merge$version.y),])
```

```
312
313   }
314
315 }
316
317 write.csv(method_add,file='method_add.csv')
318 write.csv(method_remove,file='method_remove.csv')
319
320 write.csv(android_maturity,file = 'android_maturity.csv')
321
322 # method size for all API levels
323
324 method_size <- data.frame(matrix(ncol = 23, nrow = 200))
325
326 for (j in 1:200)
327 {
328   p_size <- android_maturity[android_maturity$package == p[j],]
329
330   for (i in 1:23)
331   {
332    method_size[j,i] <- nrow(p_size[p_size$version==i,])
333
334   }
335
336 }
337
338 write.csv(method_size,file='method_size.csv')
339
340 # get the list of changed methods over all versions in the coplies APIs
341
342 setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/34_API stability
analysis/android stability")
343
344 android_stability <- read.csv("android stability input.csv", as.is =
TRUE)
345 android_stability <- android_stability %>%
group_by(class,package,method,version) %>% mutate(length(m_struct))
346 names(android_stability)[8] <- "count"
347
348 m_unique <- android_stability[android_stability$count==1,]
349 m_multiple <- android_stability[android_stability$count > 1,]
350
351 m_multiple_group <- group_by(m_multiple,class,package,method,version)
352 m_multiple_updated <- summarize(m_multiple_group,m_struct_c =
paste(m_struct, collapse = " "))
353
354 p_copied <- c("javax.sql",
355               "java.beans",
356               "java.lang.ref",
357               "java.net",
358               "java.util.logging",
359               "java.util",
360               "java.io",
361               "java.lang",
362               "java.lang.annotation",
363               "java.nio",
364               "java.nio.channels",
```

```
365                "java.nio.channels.spi",
366                "java.nio.charset",
367                "java.security",
368                "java.security.acl",
369                "java.security.cert",
370                "java.security.interfaces",
371                "java.sql",
372                "java.text",
373                "java.util.jar",
374                "java.util.prefs",
375                "java.util.zip",
376                "javax.crypto",
377                "javax.crypto.interfaces",
378                "javax.crypto.spec",
379                "javax.net",
380                "javax.security.auth",
381                "javax.security.auth.callback",
382                "javax.security.auth.login",
383                "javax.security.cert",
384                "java.nio.charset.spi",
385                "java.security.spec",
386                "java.util.regex",
387                "javax.net.ssl",
388                "javax.security.auth.x500",
389                "java.lang.reflect",
390                "java.awt.font")
391
392 # get the method change list between each API level
393
394 p_change_1 <- data.frame(class = character(),method =
character(),c_struct_pre = character(),
395                          m_struct_pre = character(),m_var_pre =
character(),version_pre = numeric(),
396                          package_pre =
character(),count_pre=numeric(),c_struct_post = character(),
397                          m_struct_post = character(),m_var_post =
character(),version_post = numeric(),
398                          package_post = character(),count_post =
numeric(),change = logical())
399 p_change_2 <- data.frame(class = character(),method =
character(),package_pre = character(),version_pre = numeric(),
400                          m_struct_pre = character(),package_post =
character(),version_post = numeric(),
401                          m_struct_post = character(),change = logical())
402
403 for (j in 1:37)
404 {
405   p_unique <- m_unique[m_unique$package ==p_copied[j],]
406   p_multiple <- m_multiple_updated[m_multiple_updated$package ==
p_copied[j],]
407
408   for (i in 1:22)
409   {
410     p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
411     p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
```

```
412     p_change_unique <- p_change_unique[p_change_unique$change == FALSE,]
413     p_change_1 <- rbind(p_change_1,p_change_unique)
414
415     p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y =
p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
416     p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
417     p_change_multiple <- p_change_multiple[p_change_multiple$change ==
FALSE,]
418     p_change_2 <- rbind(p_change_2,p_change_multiple)
419
420   }
421
422 }
423
424 write.csv(p_change_1,file='p_change_1.csv')
425 write.csv(p_change_2,file='p_change_2.csv')
426
427 # get the list of change from API level 13 to 14
428
429 android_1314 <- read.csv("13_14_input.csv", as.is = TRUE)
430 android_1314 <- android_1314 %>% group_by(class,package,method,version)
%>% mutate(length(m_struct))
431 names(android_1314)[7] <- "count"
432
433 m_unique_1314 <- android_1314[android_1314$count==1,]
434 m_multiple_1314 <- android_1314[android_1314$count > 1,]
435
436 m_multiple_1314group <-
group_by(m_multiple_1314,class,package,method,version)
437 m_multiple_1314updated <- summarize(m_multiple_1314group,m_struct_c =
paste(m_struct, collapse = " "))
438
439 p_change_1_1314 <- data.frame(class = character(),method =
character(),c_struct_pre = character(),
440                           m_struct_pre = character(),m_var_pre =
character(),version_pre = numeric(),
441                           package_pre =
character(),count_pre=numeric(),c_struct_post = character(),
442                           m_struct_post = character(),m_var_post =
character(),version_post = numeric(),
443                           package_post = character(),count_post =
numeric(),change = logical())
444 p_change_2_1314 <- data.frame(class = character(),method =
character(),package_pre = character(),version_pre = numeric(),
445                           m_struct_pre = character(),package_post =
character(),version_post = numeric(),
446                           m_struct_post = character(),change = logical())
447
448
449 for (j in 1:37)
450 {
451   p_unique1314 <- m_unique_1314[m_unique_1314$package == p_copied[j],]
452   p_multiple1314 <- m_multiple_1314updated[m_multiple_1314updated$package
== p_copied[j],]
453
```

```
454   p_change_unique1314 <- merge(x =
p_unique1314[p_unique1314$version==13,],y =
p_unique1314[p_unique1314$version==14,], by = c("class","method"),all =
FALSE)
455   p_change_unique1314$change <- p_change_unique1314$m_struct.x ==
p_change_unique1314$m_struct.y
456   p_change_unique1314 <- p_change_unique1314[p_change_unique1314$change
== FALSE,]
457   p_change_1_1314 <- rbind(p_change_1_1314,p_change_unique1314)
458
459   p_change_multiple1314 <- merge(x =
p_multiple1314[p_multiple1314$version==13,],y =
p_multiple1314[p_multiple1314$version==14,], by = c("class","method"),all =
FALSE)
460   p_change_multiple1314$change <- p_change_multiple1314$m_struct_c.x ==
p_change_multiple1314$m_struct_c.y
461   p_change_multiple1314 <-
p_change_multiple1314[p_change_multiple1314$change == FALSE,]
462   p_change_2_1314 <- rbind(p_change_2_1314,p_change_multiple1314)
463
464 }
465
466 write.csv(p_change_1_1314,file='p_change_1_1314.csv')
467 write.csv(p_change_2_1314,file='p_change_2_1314.csv')
```

PROOF OF SERVICE BY KITEWORKS

I, José E. Valdés, am over the age of eighteen years old and not a party to the within-entitled action. My place of employment and business address is Orrick, Herrington & Sutcliffe LLP, 1000 Marsh Road, Menlo Park, California 94025.

On February 29, 2016, I served the following documents:

REPLY EXPERT REPORT OF CHRIS F. KEMERER, Ph.D.

on the interested parties in this action by electronic service [Fed. Rule Civ. Proc. 5(b)] by electronically mailing a true and correct copy, pursuant to the parties agreement, to the following email addresses:

DALVIK-KVN@kvn.com
JCooper@fbm.com
gglas@fbm.com

I declare under penalty of perjury under the laws of the State of California and the United States that the foregoing is true and correct.

Executed on February 29, 2016, at San Francisco, California.

/s/ José E. Valdés
José E. Valdés